

AI can learn from data. But can it learn to reason?

Guy Van den Broeck

Outline

1. The paradox of learning to reason from data

~~deep learning~~

2. Architectures for learning and reasoning

reasoning algorithms + deep learning

Outline

1. **The paradox of learning to reason from data**



~~deep learning~~

2. Architectures for learning and reasoning

reasoning algorithms + deep learning

Can Language Models Perform Logical Reasoning?

Language Models achieve high performance on various “reasoning” benchmarks in NLP.

<p>Kristin and her son Justin went to visit her mother Carol on a nice Sunday afternoon. They went out for a movie together and had a good time.</p> 	<p>Q: How is Carol related to Justin ?</p> <p>A: Carol is the grandmother of Justin</p> 
--	---

Reasoning Example
from the CLUTRR
dataset

It is unclear whether they solve the tasks following the rules of logical deduction.

Language Models:

input → ? → *Carol is the grandmother of Justin.*

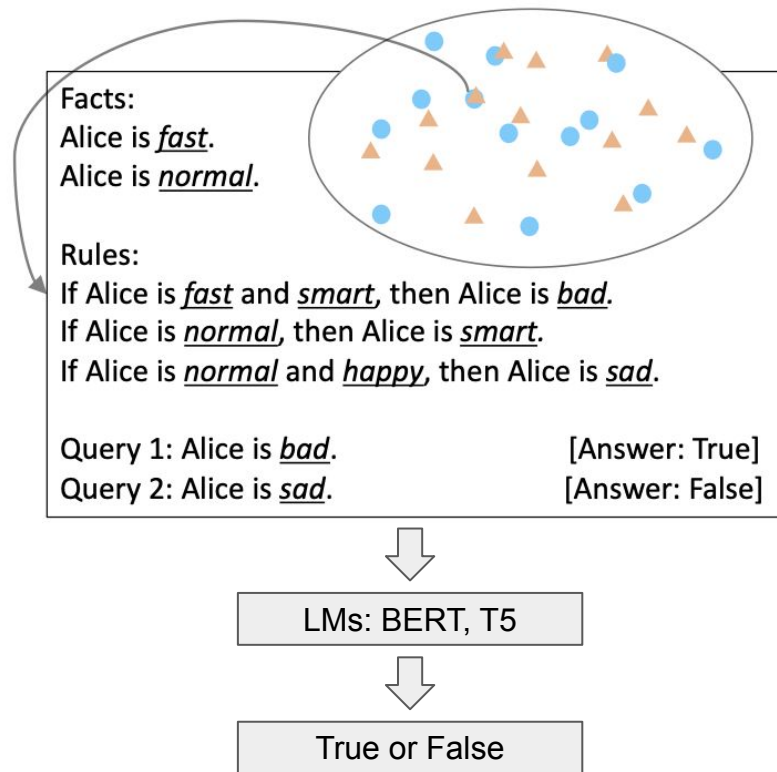
Logical Reasoning:

input → *Justin is Kristin's son; Carol is Kristin's mother;* → *Carol is Justin's mother's mother; if X is Y's mother's mother then X is Y's grandmother* → *Carol is the grandmother of Justin.*

Problem Setting: SimpleLogic

The easiest of reasoning problems:

1. **Propositional logic** fragment
 - a. bounded vocabulary & number of rules
 - b. bounded reasoning depth (≤ 6)
 - c. finite space ($\approx 10^{360}$)
2. **No language variance**: templated language
3. **Self-contained**
No prior knowledge
4. **Purely symbolic** predicates
No shortcuts from word meaning
5. **Tractable** logic (definite clauses)
Can always be solved efficiently



SimpleLogic

Generate textual train and test examples of the form:

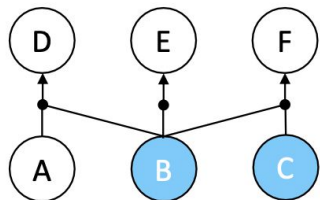
Rules: If witty, then diplomatic. If careless and condemned and attractive, then blushing. If dishonest and inquisitive and average, then shy. If average, then stormy. If popular, then blushing. If talented, then hurt. If popular and attractive, then thoughtless. If blushing and shy and stormy, then inquisitive. If adorable, then popular. If cooperative and wrong and stormy, then thoughtless. If popular, then sensible. If cooperative, then wrong. If shy and cooperative, then witty. If polite and shy and thoughtless, then talented. If polite, then condemned. If polite and wrong, then inquisitive. If dishonest and inquisitive, then talented. If blushing and dishonest, then careless. If inquisitive and dishonest, then troubled. If blushing and stormy, then shy. If diplomatic and talented, then careless. If wrong and beautiful, then popular. If ugly and shy and beautiful, then stormy. If shy and inquisitive and attractive, then diplomatic. If witty and beautiful and frightened, then adorable. If diplomatic and cooperative, then sensible. If thoughtless and inquisitive, then diplomatic. If careless and dishonest and troubled, then cooperative. If hurt and witty and troubled, then dishonest. If scared and diplomatic and troubled, then average. If ugly and wrong and careless, then average. If dishonest and scared, then polite. If talented, then dishonest. If condemned, then wrong. If wrong and troubled and blushing, then scared. If attractive and condemned, then frightened. If hurt and condemned and shy, then witty. If cooperative, then attractive. If careless, then polite. If adorable and wrong and careless, then diplomatic. Facts: Alice sensible Alice condemned Alice thoughtless Alice polite Alice scared Alice average
Query: Alice is shy ?

Training a transformer on SimpleLogic

(1) Randomly sample facts & rules.

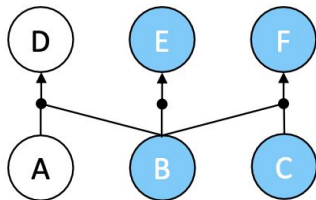
Facts: B, C

Rules: $A, B \rightarrow D$. $B \rightarrow E$. $B, C \rightarrow F$.



Rule-Priority

(2) Compute the correct labels for all predicates given the facts and rules.



Label-Priority



(1) Randomly assign labels to predicates.

True: B, C, E, F.

False: A, D.

(2) Set B, C (randomly chosen among B, C, E, F) as facts and sample rules (randomly) consistent with the label assignments.

Test accuracy for different reasoning depths

Test	0	1	2	3	4	5	6
RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5

Test	0	1	2	3	4	5	6
LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

Has the transformer learned to reason from data?

1. Easiest of reasoning problems (no variance, self-contained, purely symbolic, tractable)
2. RP/LP data covers the whole problem space
3. The learned model has almost 100% test accuracy
4. There exist transformer parameters that compute the ground-truth reasoning function:

Theorem 1: *For a BERT model with n layers and 12 attention heads, by construction, there exists a set of parameters such that the model can correctly solve any reasoning problem in SimpleLogic that requires at most $n - 2$ steps of reasoning.*

Surely, under these conditions, the transformer has learned the ground-truth reasoning function!



The Paradox of Learning to Reason from Data

Train	Test	0	1	2	3	4	5	6
RP	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
	LP	99.8	99.8	99.3	96.0	90.4	75.0	57.3
LP	RP	97.3	66.9	53.0	54.2	59.5	65.6	69.2
	LP	100.0	100.0	99.9	99.9	99.7	99.7	99.0

The BERT model trained on one distribution fails to generalize to the other distribution within the same problem space.



1. If the transformer **has learned** to reason, it should not exhibit such generalization failure.
2. If the transformer **has not learned** to reason, it is baffling how it achieves near-perfect in-distribution test accuracy.

Why? Statistical Features

Monotonicity of entailment:

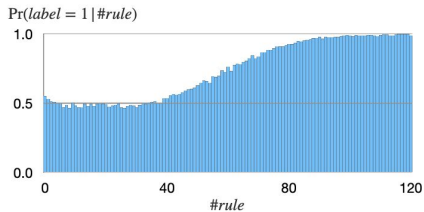
Any rules can be freely added to the axioms of any proven fact.



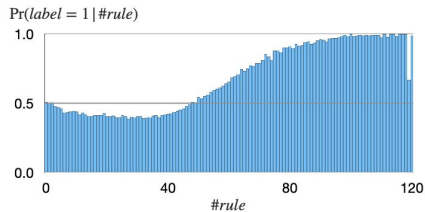
The more rules given, the more likely a predicate will be proven.



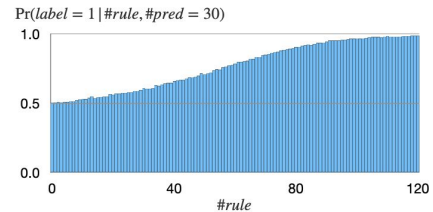
$\Pr(\text{label} = \text{True} \mid \text{Rule \#} = x)$ should increase (roughly) monotonically with x



(a) Statistics for examples generated by Rule-Priority (RP).



(b) Statistics for examples generated by Label-Priority (LP).



(c) Statistics for examples generated by uniform sampling;

Model leverages statistical features to make predictions

RP_b downsamples from RP such that $\Pr(\text{label} = \text{True} \mid \text{rule\#} = x) = 0.5$ for all x

Train	Test	0	1	2	3	4	5	6
	RP	99.9	99.8	99.7	99.3	98.3	97.5	95.5
RP	RP_b	99.0	99.3	98.5	97.5	96.7	93.5	88.3

1. Accuracy drop from RP to RP_b indicates that **the model is using rule# as a statistical feature to make predictions.**
2. Potentially countless statistical features
3. Such features are **inherent to the reasoning problem**, cannot make data “clean”

First Conclusion

Experiments unveil the fundamental difference between

1. learning to reason, and
2. learning to achieve high performance on benchmarks using statistical features.

Be careful deploying AI in applications where this difference matters.

FAQ: Do bigger transformers solve this problem? No, already 99% accurate...

FAQ: Will reasoning emerge? Perhaps on 99% of human behavior...

Outline

1. The paradox of learning to reason from data

~~deep learning~~

2. **Architectures for learning and reasoning**

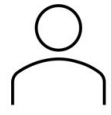
reasoning algorithms + deep learning

Generative models are still hard to control

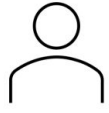
more reasoning!

Generate image





Generate a sentence using "frisbee", "caught" and "dog", following the given order.

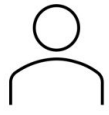


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

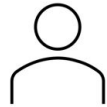


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

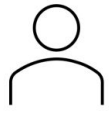


That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT

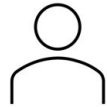


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT



That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT



A frisbee is caught by a dog.

A pair of frisbee players are caught in a dog fight.

GeLaTo

What do we have?

Prefix: “The weather is”

Constraint α : text contains “winter”

Model only does $p(\text{next-token}|\text{prefix}) =$

cold	0.05
warm	0.10

Train some $q(.|\alpha)$ for a specific task distribution $\alpha \sim p_{\text{task}}$
(*amortized inference, encoder, masked model, seq2seq, prompt tuning,...*)

Train $q(\text{next-token}|\text{prefix}, \alpha)$

What do we need?

Prefix: “The weather is”

Constraint α : text contains “winter”

Generate from $p(\text{next-token}|\text{prefix}, \alpha) =$

cold	0.50
warm	0.01

$$\propto \sum_{\text{text}} p(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$

Marginalization!

Tractable Probabilistic Models

Tractable Probabilistic Models (TPMs) model **joint probability distributions** (just like auto-regressive LMs) and allow **efficient** computation of various probabilistic queries.

e.g., efficient marginalization:

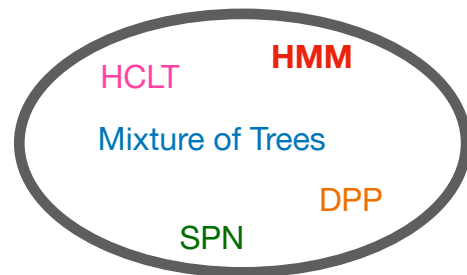
$$p_{\text{TPM}}(\text{3rd token} = \text{frisbee}, \text{5th token} = \text{dog})$$

in particular ...

$$\sum_{\text{sentence}} p_{\text{TPM}}(\text{sentence}, \text{next-token} = \text{"warm"}, \text{prefix} = \text{"The weather is"}, \alpha)$$

➔ Efficient conditioning given lexical constraints : $p_{\text{TPM}}(\text{next-token} \mid \text{prefix}, \alpha)$

Probabilistic (Generating) Circuits



Step 1: Distill an HMM p_{hmm} that approximates p_{gpt}



1. An HMM with 4096 hidden states and 50k emission tokens
2. Train the HMM on data sampled from GPT2-large (domain-adapted, either via prompting or fine-tuning), effectively minimizing $\text{KL}(p_{\text{gpt}} // p_{\text{HMM}})$
3. Leverages the latent variable distillation technique for training probabilistic circuits at scale [ICLR 23]. (Cluster embeddings of examples to estimate latent Z_i)

Computing $p_{\text{hmm}}(\alpha \mid x_{1:t+1})$

For α in conjunctive normal form (CNF):

$$(w_{1,1} \vee \dots \vee w_{1,d_1}) \wedge \dots \wedge (w_{m,1} \vee \dots \vee w_{m,d_m})$$

where each w_{ij} is a keyword (i.e. a string of tokens),
representing the constraint that w_{ij} appears in the generated text.

e.g., $\alpha = (\text{"swims"} \vee \text{"like swimming"}) \wedge (\text{"lake"} \vee \text{"pool"})$

Efficient algorithm:

For m clauses and sequence length n , time-complexity for generation is $O(2^m n)$.

Trick: dynamic programming with clever preprocessing and local belief updates

CommonGen: a Challenging Benchmark

Given 3-5 concepts (keywords), our goal is to generate a sentence using all keywords, which can appear in any order and any form of inflections. e.g.,

Input: snow drive car

Reference 1: A car drives down a snow covered road.

Reference 2: Two cars drove through the snow.

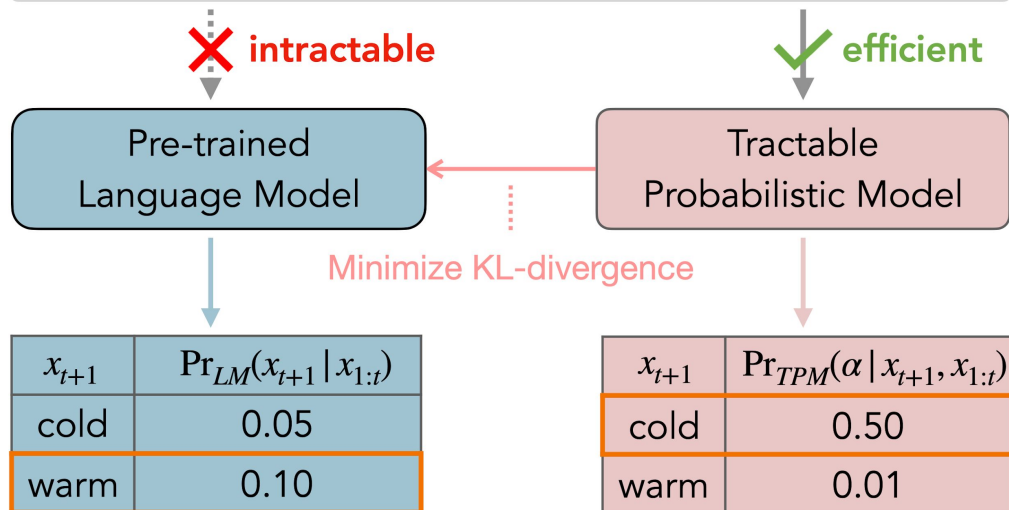
$$(w_{1,1} \vee \dots \vee w_{1,d_1}) \wedge \dots \wedge (w_{m,1} \vee \dots \vee w_{m,d_m})$$

Each clause represents the inflections for one keyword.

GeLaTo Overview

Lexical Constraint α : sentence contains keyword "winter"

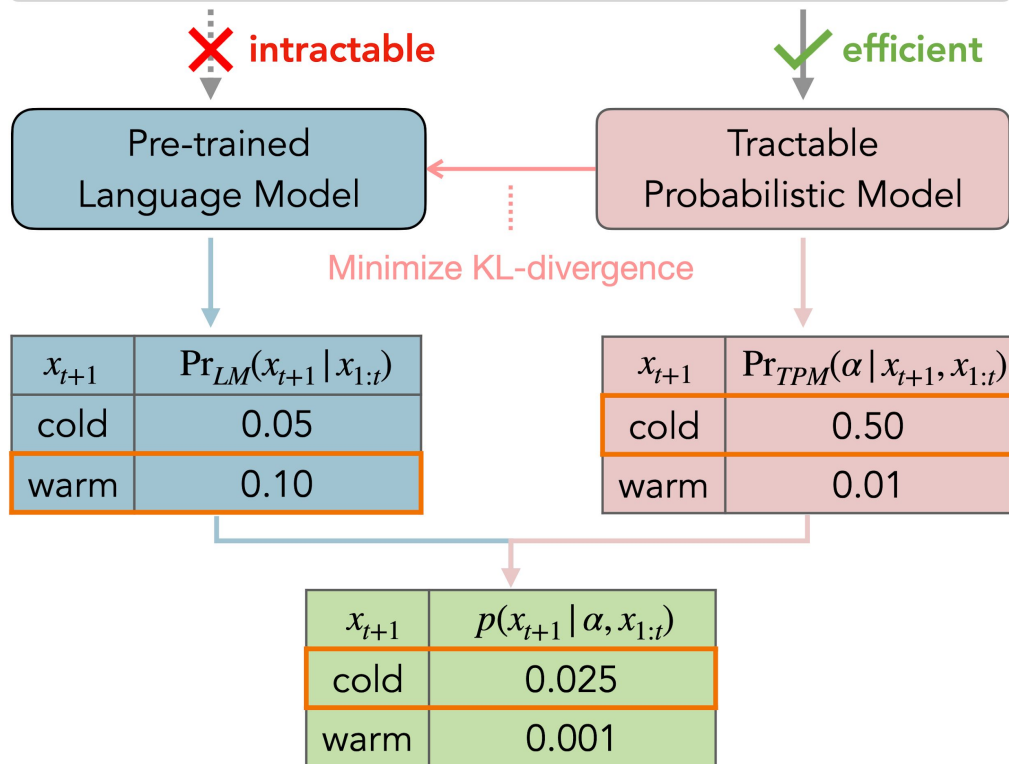
Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$



GeLaTo Overview

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$



Step 2: Control p_{gpt} via p_{hmm}

Unsupervised

Language model is not fine-tuned/prompted to satisfy constraints

By Bayes rule:

$$p_{gpt}(x_{t+1} | x_{1:t}, \alpha) \propto p_{gpt}(\alpha | x_{1:t+1}) \cdot p_{gpt}(x_{t+1} | x_{1:t})$$

Assume $p_{hmm}(\alpha | x_{1:t+1}) \approx p_{gpt}(\alpha | x_{1:t+1})$, we generate from:

$$p(x_{t+1} | x_{1:t}, \alpha) \propto p_{hmm}(\alpha | x_{1:t+1}) \cdot p_{gpt}(x_{t+1} | x_{1:t})$$

Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
<i>Unsupervised</i>												
InsNet (Lu et al., 2022a)	-	-	18.7	-	-	-	-	-	100.0	-	100.0	-
NeuroLogic (Lu et al., 2021)	-	41.9	-	24.7	-	14.4	-	27.5	-	96.7	-	-
A*esque (Lu et al., 2022b)	-	44.3	-	28.6	-	15.6	-	29.6	-	97.1	-	-
NADO (Meng et al., 2022)	-	-	26.2	-	-	-	-	-	96.1	-	-	-
GeLaTo	44.6	44.1	29.9	29.4	16.0	15.8	31.3	31.0	100.0	100.0	100.0	100.0

Step 2: Control p_{gpt} via p_{hmm}

Supervised

Language model is fine-tuned to perform constrained generation (e.g. seq2seq)

Empirically $p_{HMM}(\alpha | x_{1:t+1}) \approx p_{gpt}(\alpha | x_{1:t+1})$
does not hold well enough;

we view $p_{HMM}(x_{t+1} | x_{1:t}, \alpha)$ and $p_{gpt}(x_{t+1} | x_{1:t})$ as classifiers trained for the same task with different biases; thus we generate from their weighted geometric mean:

$$p(x_{t+1} | x_{1:t}, \alpha) \propto p_{hmm}(x_{t+1} | x_{1:t}, \alpha)^w \cdot p_{gpt}(x_{t+1} | x_{1:t})^{1-w}$$

Method	Generation Quality								Constraint Satisfaction			
	ROUGE-L		BLEU-4		CIDEr		SPICE		Coverage		Success Rate	
	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>	<i>dev</i>	<i>test</i>
<i>Supervised</i>												
NeuroLogic (Lu et al., 2021)	-	42.8	-	26.7	-	14.7	-	30.5	-	97.7	-	93.9 [†]
A*esque (Lu et al., 2022b)	-	43.6	-	28.2	-	15.2	-	30.8	-	97.8	-	97.9 [†]
NADO (Meng et al., 2022)	44.4 [†]	-	30.8	-	16.1 [†]	-	32.0[†]	-	97.1	-	88.8 [†]	-
GeLaTo	46.0	45.6	34.1	32.9	16.7	16.8	31.3	31.9	100.0	100.0	100.0	100.0

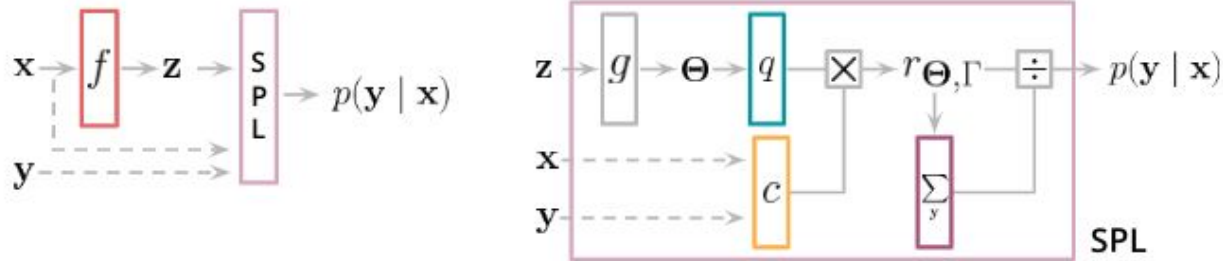
Advantages of our framework:

1. Constraint α is guaranteed to be satisfied: for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} | x_{1:t}, \alpha) = 0$ for both settings.
2. Training p_{hmm} does not depend on α , which is only imposed at inference (generation) time. Once p_{hmm} is trained, we can impose whatever α .
3. We can impose additional tractable constraints:
 - The keywords are generated following a particular order.
 - (Some) keywords must appear at a particular position.
 - (Some) keywords must not appear in the generated sentence.

Conclusion: you can control an intractable generative model using a tractable generative model for (symbolic) reasoning.

Dessert: Semantic Probabilistic Layers

- How to give a 100% guarantee that Boolean constraints will be satisfied?
- Bake the constraint into the neural network as a special layer



- Secret sauce is *tractable circuits* – computation graphs for reasoning



GROUND TRUTH



RESNET-18



SEMANTIC LOSS



SPL (ours)

ARCHITECTURE

EXACT MATCH

HAMMING SCORE

CONSISTENCY

*Is prediction
a shortest path?*

**This is the real
task!**

*Are individual
edge predictions
correct?*

*Is output
a path?*



GROUND TRUTH



RESNET-18



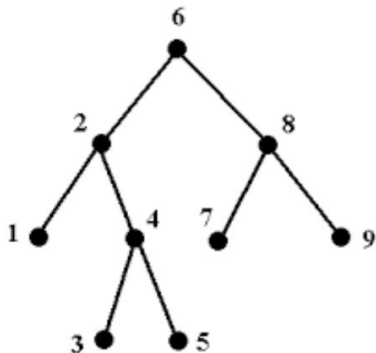
SEMANTIC LOSS



SPL (ours)

ARCHITECTURE	EXACT MATCH	HAMMING SCORE	CONSISTENCY
RESNET-18+FIL	55.0	97.7	56.9
RESNET-18+ \mathcal{L}_{SL}	59.4	97.7	61.2
RESNET-18+SPL	75.1	97.6	100.0
OVERPARAM. SDD	78.2	96.3	100.0

Hierarchical Multi-Label Classification

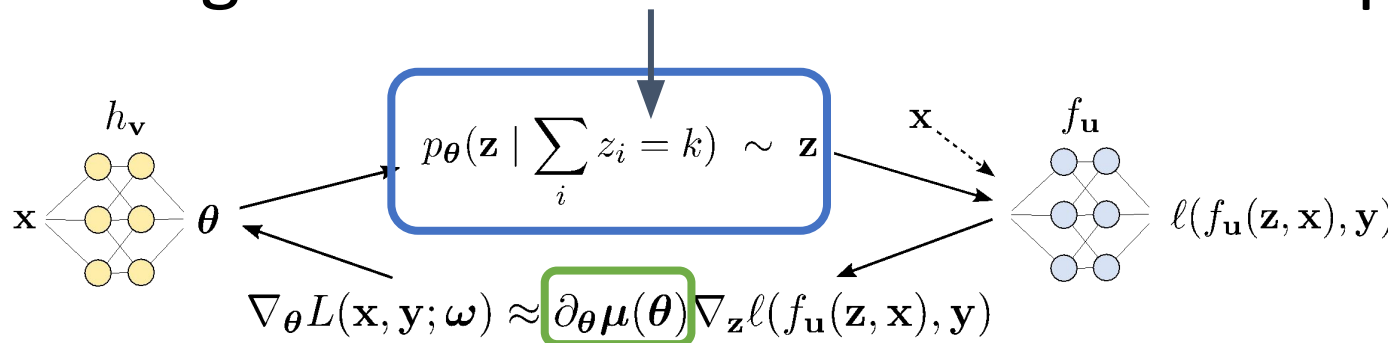


“if the image is classified as a dog, it must also be classified as an animal”

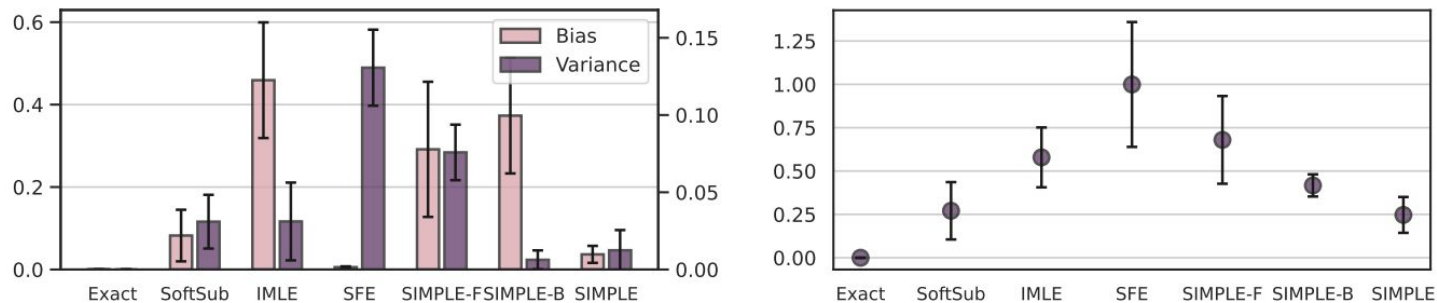
“if the image is classified as an animal, it must be classified as either cat or dog”

DATASET	EXACT MATCH	
	HMCNN	MLP+SPL
CELLCYCLE	3.05 ± 0.11	3.79 ± 0.18
DERISI	1.39 ± 0.47	2.28 ± 0.23
EISEN	5.40 ± 0.15	6.18 ± 0.33
EXPR	4.20 ± 0.21	5.54 ± 0.36
GASCH1	3.48 ± 0.96	4.65 ± 0.30
GASCH2	3.11 ± 0.08	3.95 ± 0.28
SEQ	5.24 ± 0.27	7.98 ± 0.28
SPO	1.97 ± 0.06	1.92 ± 0.11
DIATOMS	48.21 ± 0.57	58.71 ± 0.68
ENRON	5.97 ± 0.56	8.18 ± 0.68
IMCLEF07A	79.75 ± 0.38	86.08 ± 0.45
IMCLEF07D	76.47 ± 0.35	81.06 ± 0.68

Reasoning about constraints in the latent space



SIMPLE achieves **lower bias and variance** by **exact, discrete samples** and **exact derivative of conditional marginals**.



and SotA Learning to Explain (L2X) and sparse discrete VAE results.

Outline

1. The paradox of learning to reason from data

~~*deep learning*~~

2. Architectures for learning and reasoning

reasoning algorithms + deep learning

Thanks

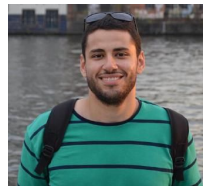
This was the work of many wonderful students/postdocs/collaborators!



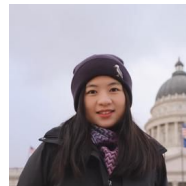
Honghua



Meihua



Kareem



Zhe

References: <http://starai.cs.ucla.edu/publications/>